Differential gene expression analysis

The aim of this task is to determine, which genes are differentially regulated based on treatment exposure.

**Data files for the analysis:**

1. Covariates file (n = 60 individuals), including sex, age, body-mass index (bmi), treatment, batch effects, and sex. Batch effects occur because measurements are affected by laboratory conditions, reagent lots, and personnel differences. This becomes a major problem when batch effects are confounded with an outcome of interest and lead to incorrect conclusions. In our data batches are codes as: batch 1 = chip (n=10) batch2 = plate column (n=12) batch 3 = plate (n=2, a plate has 96 positions 12 columns and 8 rows. Rows are equal to chip)
2. Gene expression files (n = 47298 features array probes of the Illumina Human HT-12v4 Bead Chip, 120 samples (60 individuals at baseline (treatment=0) and after treatment (=1))).

Before we start, the working directory should be set, libraries necessary for data analysis and manipulation attached, data files loaded:

rm(list=ls())  
setwd("C:/Users/SVITLANA/Desktop/PhD Application/Assignment")  
  
library(beadarray)  
library(limma)  
library (MASS)  
  
load("expression\_data.rda")  
pheno\_data <- read.table("pheno\_dat.txt")  
colnames(pheno\_data) <- sapply(pheno\_data[1,], as.character)  
pheno\_data <- pheno\_data[-1,]

## Assignment

*Step 1*: Apply a quality control (QC) on the gene expression data including: a) filtering by a detection p-value (0.05 in 50% of the samples) b) normalization c) batch correction

*Step 2*: Apply QC on the phenotype data (covariates) including outlier filtering.

*Step 3*: How many significantly differently regulated genes do you find and how many array probes do they comprise?

*Step 4*: How many of the detected genes (from Step 2) have a fold change 1.2?

## Solution

**Step 1 (a)**

Filtering non-responding probes from further analysis can improve the power to detect differential expression. One way of achieving this is to remove probes whose probe sequence has undesirable properties.

Since the detection p-values are already available within the “expression\_data.rda” file (which is stored as BSData) in the “assayData” object, we may use them to perform filtering procedure.

For every probe, we check how many samples are statistically signifficant by selecting only those detection values which are smaller than 0.05. If the amount of signifficant samples is smaller than half of the number of samples, the probe is filtered out of the data.

assay.Data <- attr(BSData,"assayData")  
  
filt <- apply(assay.Data$Detection < 0.05, 1, sum) >= dim(assay.Data$Detection)[2]\*0.5  
  
BSData.filt <- BSData[filt,]

After filtering procedure, the number of probes decreased from 47298 to 15711:

dim(BSData)

## Features Samples Channels   
## 47298 120 1

dim(BSData.filt)

## Features Samples Channels   
## 15711 120 1

**Step 1 (b)**

To correct for differences in expression level across a chip and between chips we need to normalise the signal to make the arrays comparable.

The scanner generally produces values for the probe-specific average intensities in the range 0 to 2^16 - 1, although the image manipulation and background subtraction steps can lead to values outside this range. This is not a convenient scale for visualization and analysis and it is common to convert intensities onto the approximate range 0 to 16 using a log transformation.

We apply quantile normalization method to the log-transformed data using the *normaliseIllumina* function from *beadarray* package.

BSData.norm <- normaliseIllumina(BSData.filt, method="quantile", transform="log2")

**Step 1 (c)**

Firstly, we have to to merge expression data with phenotype data:

temp <- merge(pData(BSData.norm),   
 pheno\_data, by.x = "sampleID", by.y = "ID")  
pData(BSData.norm) <- temp

Secondly, the phenotype (covariates) data has missing values. Since there is only one individual with missing information about batch effects, age, sex, treatment and BMI, we can remove this individual from the data-set.

missing <- apply(is.na(pData(BSData.norm)),1,sum) > 0  
  
pData(BSData.norm)[which(missing),"individual\_ID"]

## [1] individual\_60 individual\_60  
## 61 Levels: individual\_1 individual\_10 individual\_11 ... individual\_ID

BSData.norm.clean <- BSData.norm[,!missing]

Now, we need to repeat filtering for cleaned data:

assay.Data.clean <- attr(BSData.norm.clean,"assayData")  
filt.clean <- apply(assay.Data.clean$Detection < 0.05, 1, sum) >= dim(assay.Data.clean$Detection)[2]\*0.5  
BSData.filt.clean <- BSData.norm.clean[filt.clean,]

The number of probes slightly decreased when we removed the observation with missing values:

dim(BSData.norm.clean)

## Features Samples Channels   
## 15711 118 1

dim(BSData.filt.clean)

## Features Samples Channels   
## 15682 118 1

Next, we remove batch effects from intensities sobsequently for every batch:

assay.Data.filt.clean <- attr(BSData.filt.clean,"assayData")  
assay.Data.filt.clean.batch1Corrected <- removeBatchEffect(assay.Data.filt.clean$exprs,   
 batch = BSData.filt.clean$batch1)  
assay.Data.filt.clean.batch12Corrected <-   
 removeBatchEffect(assay.Data.filt.clean.batch1Corrected,   
 batch = BSData.filt.clean$batch2)  
  
assay.Data.filt.clean.batch123Corrected <-   
 removeBatchEffect(assay.Data.filt.clean.batch12Corrected,   
 batch = BSData.filt.clean$batch3)  
  
assay.Data.corrected <- assay.Data.filt.clean.batch123Corrected

assay.Data.filt.clean$exprs[1:5,1:5]

## 9992555076\_A 9992555076\_B 9992555076\_C 9992555076\_D 9992555076\_E  
## ILMN\_1698554 6.503347 6.679735 6.813419 6.612795 6.947694  
## ILMN\_1814092 6.599601 6.568393 6.902263 6.642158 6.576087  
## ILMN\_2061446 8.091643 8.100258 7.740154 7.823720 7.663109  
## ILMN\_1676336 6.963527 6.714372 6.842718 6.845546 6.724543  
## ILMN\_3237396 8.171547 7.603882 7.922723 7.782136 7.772067

assay.Data.corrected[1:5,1:5]

## 9992555076\_A 9992555076\_B 9992555076\_C 9992555076\_D 9992555076\_E  
## ILMN\_1698554 6.593506 6.691423 6.815443 6.611305 6.898706  
## ILMN\_1814092 6.325682 6.334164 6.739891 6.420554 6.483519  
## ILMN\_2061446 7.860004 7.855882 7.545246 7.690278 7.422643  
## ILMN\_1676336 6.928724 6.632153 6.837250 6.542590 6.658281  
## ILMN\_3237396 8.046848 7.589758 8.014824 8.011922 7.719490

**Step 2**

There various techniques how to perform model diagnostics and identify outliers. The unusual values which do not follow the norm are called an outlier. Outliers present a particular challenge for analysis, and thus it becomes essential to identify these values and tackle them.

We are interested if there are any outlier values among covariates. As only *age* and *bmi* are continuous variables, there is no point of analysing other predictor variables for outliers. Prepare the variables of interest:

treatment <- BSData.filt.clean$treatment  
sex <- BSData.filt.clean$sex  
age <- BSData.filt.clean$age  
bmi <- BSData.filt.clean$bmi  
  
age\_num <- as.numeric(as.character(age))  
sex\_num <- as.numeric(as.character(sex))  
treatment\_num <- as.numeric(as.character(treatment))  
bmi\_num <- as.numeric(as.character(bmi))

Leverage is a measure of how far away the independent variable values of an observation are from those of the other observations. This metric is related to covariates only and does not comprise for possible association with outcome variable.

X <- as.matrix(data.frame(age\_num, bmi\_num))  
  
Xt <- t(X)  
W <- Xt %\*% X  
det(W)!=0

## [1] TRUE

Hence, inverse matrix exists and therefore we may compute the hat matrix:

W\_inv <- ginv(W)  
H <- X %\*% W\_inv %\*% Xt  
leverage <- ifelse(diag(H)>4/118,"high","not high")  
  
outlier <- leverage == "high"  
  
BSData.final <- BSData.filt.clean[,!outlier]  
dim(BSData.final)

## Features Samples Channels   
## 15682 112 1

## remove outliers from assayData:  
assay.Data.corrected <- assay.Data.corrected[,!outlier]  
dim(assay.Data.corrected)

## [1] 15682 112